Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gregor Brajar

Ogrodje Kivy za hiter razvoj mobilnih aplikacij

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Alenka Kavčič

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Ogrodje Kivy za hiter razvoj mobilnih aplikacij

Tematika naloge:

V okviru diplomske naloge preučite ogrodje Kivy, ki je namenjeno hitremu razvoju aplikacij, tako mobilnih kot namiznih, in omogoča enostavno uporabo naravnih uporabniških vmesnikov. Opišite glavne značilnosti ogrodja Kivy ter ga primerjajte z drugimi ogrodji za razvoj platformno neodvisnih mobilnih aplikacij. Izdelajte tudi enostavno mobilno aplikacijo, ki vključuje uporabo senzorjev mobilne naprave, ter jo pripravite za izvajanje na več različnih platformah (npr. Android in iOS).

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Gregor Brajar sem avtor diplomskega dela z naslovom:

Orodje Kivy za hiter razvoj mobilnih aplikacij

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Alenka Kavčič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 14. marec 2016 Podpis avtorja:

Rad bi se zahvalil svoji mentorici viš. pred. dr. Alenki Kavčič za strokovno svetovanje, nasvete in pripombe. Zahvala gre tudi očetu Antonu in mami Anici, bratoma Primožu in Matjažu, prijateljem Bojanu Staniši, Evi Kremesec, Klari Golić, Ireni Ham, Tomažu Vidicu, Rafkotu in Darinki Krevs in vsem drugim, ki sem jih nehote izpustil. Hvala vam za vse spodbudne besede.

”Mirno morje še nikoli ni naredilo dobrega mornarja!”

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
OS	operating system	Operacijski sistem
UV	user interface	Uporabniški vmesnik
API	Application programming interface	Programski vmesnik
HTML	Hyper Text Markup Language	Označevalni jezik za oblikovanje večpredstavnostnih dokumentov
CSS	Cascading Style Sheets	Prekrivni slogi
PHP	Hypertext Preprocessor	Programski jezik za izdelavo dinamičnih spletnih strani
SDK	Software development kit	Nabor razvojnih programskih orodij
NDK	Native development kit	Orodje, ki omogoča izvajanje posameznih delov kode, napisanih v jeziku C in C++ znotraj aplikacije Android
OpenGl	Open Graphics Library	Programski vmesnik za pisanje rač - unalniških programov, ki prikazujejo 2D in 3D računalniško grafiko

Povzetek

Na trgu je vse več orodij in knjižnic, ki ponujajo razvoj aplikacij za več platform. To je posledica povečanega števila mobilnih operacijskih sistemov. Cilj diplomskega dela je spoznati in preizkusiti Kivy, ki ponuja razvoj aplikacij za namizne in mobilne naprave. Orodje smo primerjali z dvema orodjema, PhoneGap in Titanium, ki sta danes na trgu konkurenčna. Kivy je knjižnica za Python. Do vseh funkcionalnosti naprave dostopa preko knjižnic. Pri izdelavi mobilne aplikacije sem spoznal tudi jezik KV, s katerim opišemo grafični uporabniški vmesnik. Pri razvoju aplikacije smo uporabili tudi modul, ki omogoča transformacije nad objektom, to so operacije premik, rotacija in povečanje/zmanjšanje, ter knjižnico za nadzor in uporabo senzorjev mobilne naprave, v našem primeru kamere.

Ključne besede: orodje, Kivy, knjižnjica, medplatformski razvoj, mobilna aplikacija, namizna aplikacija.

Abstract

There are more and more tools and libraries, offering multi-platform application development, on the market. This is due increased number of mobile operating systems in the last couple of years. The goal of this thesis is to get familiar with Kivy architecture, which provides support for desktop and mobile application development. We compared the mentioned tool with competing tools called PhoneGap and Titanium. Kivy is a library for Python programming language. All of the functions of the device are accessed via libraries. Through development process we learned the KV programming language, which we used to describe/develop graphical user interface. We also used a modul, which supports transformations of the objects, that is translation, rotation and scale, and library which supports access and control of the mobile device sensors (in our case the camera).

Keywords: works, Kivy, libraries, cross platform developement, mobile application, desktop application.

Poglavje 1

Uvod

Mobilni svet se v današnjem času hitro razvija in temelji na zmogljivejših telefonih, njihovem dizajnu[1], operacijskem sistemu in tudi strojni opremi. Osnovni funkciji (tj. zvočni pogovor), ki je bila zasnovana na začetku izuma mobilnega telefona leta 1978, so se do danes pridružile še številne druge dodatne storitve. Skupne storitve, kot so npr. pošiljanje kratkih besedilnih sporočil (SMS-sporočila), fotografiranje, predvajanje glasbe, uporaba interneta in spletne pošte, uporaba navigacije, »pospeškometra« ipd., so danes zelo priljubljene in zaželene.

Možnosti za uporabo mobilnih naprav so tako s časom vse večje. Te naprave omogočajo uporabnikom, da opravijo čedalje več nalog z uporabo pametnih telefonov ali tabličnega računalnika. Tako se podjetja osredotočajo tudi na izdelavo mobilnih aplikacij, ki so enostavne in hitro pripravljene za uporabo na trgu. Namen teh aplikacij je različen – nekatere služijo za prosti čas, druge kot poslovne aplikacije. Na trgu se lahko nudijo brezplačno (npr. aplikacije Facebook, Youtube in Petrol) oziroma se zaračunavajo (npr. aplikacije Navigon, MotoGP in Live Experience). Zaradi razširjenosti trga izdelava mobilnih aplikacij niti ni tako enostavna, saj je hitra in stalna rast mobilne tehnologije prinesla razvoj mnogih platform. Obstoj različnih mobilnih operacijskih

[1]Pametni telefoni so prijetno oblikovani in enostavni za uporabo. Pomembni so velikost in debelost ohišja, razporeditev itd.

sistemov z različnimi programskimi jeziki in razvojnimi orodij je tako povzročil problem pri razvoju aplikacij za več platform. Programiranje za vsak operacijski sistem posebej zahteva veliko znanja in časa, razvoj pa je drag ter zamuden. Veliko aplikacij uporablja iste funkcionalnosti, a na različnih mobilnih sistemih. V času študija sem opravil kar nekaj razgovorov za delo prek študentskega servisa, kjer sem nekajkrat naletel na isto vprašanje, in sicer kako bi pri razvoju aplikacije iste funkcionalnosti definiral samo enkrat, pri vsaki naslednji aplikaciji pa uporabil že narejeno. Statistika najbolj uporabljenih aplikacij kaže, da v sam vrh sodijo sledeče aplikacije: Facebook, Messanger, Instagram, Twitter, Email, Snapchat. Vse te aplikacije so dostopne na vseh treh najbolj uporabnih mobilnih sistemih (Android, iOS in Windows Phone). Skupne lastnosti teh aplikacij so uporaba kamere za zajem slike, snemanje in urejanje slik ter druge lastnosti. Na podlagi navedenega smo se tako v svojem delu osredotočili na eno izmed orodij, v katerem smo te funkcionalnosti najprej implementirali, nato pa uporabili v različnih sistemih. Knjižnica, ki smo jo pri tem uporabili, se imenuje Kivy. Uporaba kamere se deli na dva dela. Sliko zajamemo s fotoaparatom ali pa jo izberemo iz galerije. To smo tudi implementirali. Velika prednost te knjižnice je že dodana implementacija za premik slik in obračanje oziroma povečavo.

Poglavje 2

Razvoj mobilnih aplikacij

Za razvoj mobilne aplikacije lahko uporabimo tri različne pristope. Ti pristopi so:

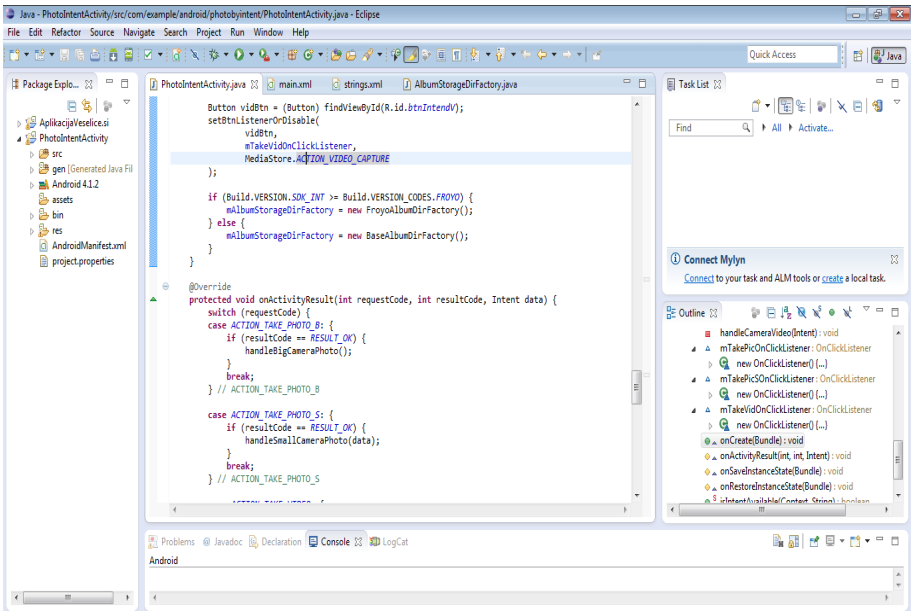
- avtohtona (ang. *Native*) aplikacija, z uporabo ustreznega programskega jezika
- spletna aplikacija, ki se izvaja preko spletnega brskalnika z uporabo spletnih tehnologij
- uporaba ogrodja ali knjižnice, ki omogoča razvoj ene kode in jo potem ogrodje prilagodi konkretni izvajalni platformi (hibridne aplikacije)

Hitro spreminjanje tehnologij in nove hitro razvijajoče se platforme povzročajo težave pri razvoju aplikacij. Razvoj za sistem Android ima več svobode, kar pomeni manj omejitev. Platforma je bolj fleksibilna in delno odprtokodna. Aplikacije za iOS je lažje pisati, saj je podana ena platforma. Aplikacijski vmesnik in programske knjižnice so bolj dodelane.

2.1 Avtohtone aplikacije

Mobilni svet ponuja različne izbire operacijskega sistema. Ta pa zahteva določen programski jezik. Že pri najpogostejših treh operacijskih sistemih

po uporabi naletimo na neenakost. Tako je za Android določen programski jezik Java, za razvoj aplikacij iOS se uporablja programski jezik Objective-C in za razvoj Windows Phone aplikacij programski jezik C#. Za vsako platformo so pripravljene knjižnice in orodja, ki olajšajo delo. Na spletu so tudi pripravljena razvojna okolja. Eno izmed njih prikazuje slika 2.1, ki kaže primer razvoja v jeziku Java. Aplikacija, ki je spisana v programskem jeziku, je narejena za določeno strojno opremo in za njen operacijski sistem. Tej vrsti aplikacije pravimo avtohtona mobilna aplikacija. Aplikacija je neposredno nameščena na napravo in jo uporabniki običajno dobijo preko spletne trgovine.



Slika 2.1: Primer avtohtone aplikacije v Eclipsu.

2.1.1 Prednosti in slabosti

Avtohtone aplikacije nudijo hitro delovanje in visoko stopnjo zanesljivosti. Omogočajo tudi dostop do različnih naprav telefona, nekatere aplikacije pa lahko uporabniki uporabljajo brez internetne povezave [1]. Primerne so tudi

za razvoj video iger. V primeru, da se podjetje odloči za razvoj aplikacije, ki bi bila združljiva z drugimi napravami in operacijskimi sistemi, je to lahko kar velik strošek. Na podlagi zgoraj navedenega se lahko sklepa, da samo vzdrževanje predstavlja dodaten strošek.

2.2 Spletne aplikacije

V zadnjih nekaj letih je internet zelo napredoval. S tem so se pojavile tudi tehnologije, ki hkrati omogočajo videz in uporabnost (npr. spletne strani). To pa omogočajo sledeče tehnologije:

- HTML (ang. *Hyper Text Markup Language*)
- CSS (ang. *Cascading Style Sheets*)
- JavaScript
- PHP

Z jezikom HTML omogočimo prikaz dokumenta v spletnem brskalniku. Z zaporedjem ukazov povemo, kako naj se določena stran pokaže, CSS uporabimo za oblikovanje spletne strani, JavaScript pomaga pri izdelavi interaktivnih spletnih strani, s programskim jezikom PHP pa je možno razviti dinamično spletno stran. Slika 2.2 prikazuje kodo HTML. Z dodanim elementom <link> v kodi HTML povemo brskalniku, da se koda CSS nahaja v datoteki z imenom "datoteka.css". Prav tako se koda JavaScript nahaja v zunanji datoteki, kar je nakazano z elementom <script>. Aplikacija, razvita s spletnimi tehnologijami, je neodvisna od platforme in enostavna, saj uporablja interakcijo odjemalec–strežnik. JavaScript se izvaja na odjemalcu, PHP pa na strežniku.

2.2.1 Prednosti in slabosti

Spletne aplikacije je lažje vzdrževati, saj imajo skupno kodo za več mobilnih sistemov [2]. Uporabnik ne rabi skrbeti za posodobitve. Za namestitev

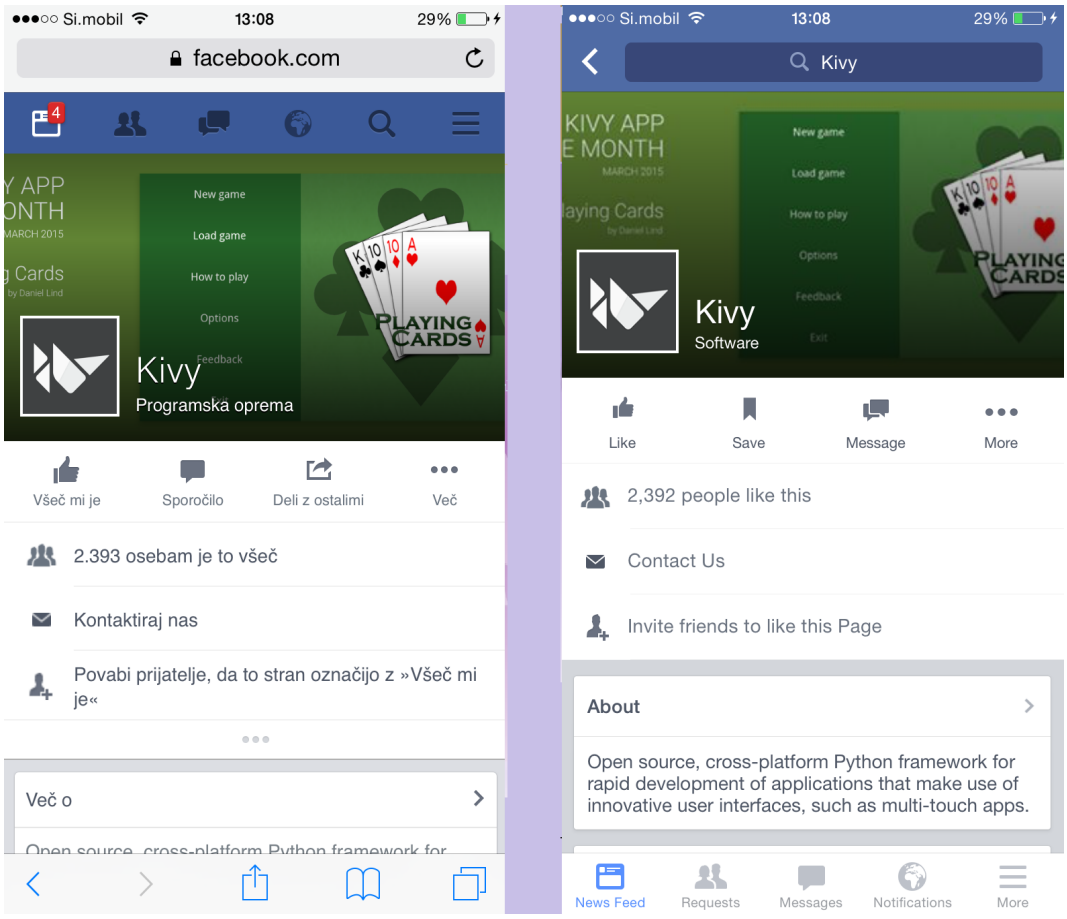
```
<!DOCTYPE html>
<html>
  <head>
    <title>jQuery - Predmeti</title>
    <link rel="stylesheet" type="text/css" href="datoteka.css">
    <script src="jquery.js"></script>
    <script src="skripta2.js"></script>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <div id="sredina2">
      <h2>2. naloga</h2>
      <h3>Izberi predmete</h3>
      <div>
        "
                    Predmeti:"
        <br>
        <select id="predmeti" size="10">…</select>
        <br>
        <input type="button" id="dodaj" value="Dodaj">
      </div>
      <div>…</div>
    </div>
  </body>
</html>
```

Slika 2.2: Primer kode HTML za prikaz spletne aplikacije.

aplikacije ni potreben obisk trgovine z aplikacijami. Za njeno uporabo pa je potreben internet. Obstaja več mobilnih brskalnikov, ki razvijalcem predstavljajo večji strošek razvijanja in vzdrževanja. Aplikacije so manj privlačne od izvornih, saj niso nameščene v napravi. Pri uporabi senzorjev smo bolj omejeni, predvsem pri uporabi strojne (ang. *Hardware*) podpore. Tako ni mogoč dostop do npr. magnetometra, tlačnega senzorja, senzorjev bližine, svetlobe, temperature, vibracij, stanja baterij, obvestil in podobnih [3, 4].

2.2.2 Kaj izbrati?

Razvoj avtohtone ali spletne aplikacije prinaša tako dobre kot tudi slabe lastnosti. Kot bomo videli v nadaljevanju, so razlike v razvojnem jeziku, stroških razvoja in vzdrževanja in tudi v načinu dostopa do aplikacije.



Slika 2.3: Videz mobilne aplikacije kot spletna (levo) in avtohtona (desno).

Slika 2.3 prikazuje videz dveh vrst aplikacij. Primer je prikazan na sistemu iOS. Prva stvar pri mobilnem razvoju aplikacije je izgled uporabniškega vmesnika. Na prvi pogled je videti, da bistvene razlike ni.
Razvoj avtohtone mobilne aplikacije zahteva svoj razvojni proces. Vsaka platforma ima svoj avtohtoni programski jezik: Java (Android), Objektni C

(iOS), Visual C# (Windows Phone) ipd. Razvoj in samo vzdrževanje povzročata relativno visok strošek, mobilna spletna aplikacija pa teče v spletnem brskalniku naprave [5]. Razvoj poteka v sledečih jezikih: HTML, CSS in JavaScript.

Avtohtono mobilno aplikacijo je najprej treba prenesti na mobilno napravo, da lahko deluje kot samostojna aplikacija. Vse posodobitve mora uporabnik sam ročno prenesti in namestiti. Za posodobitev se lahko odloči ali pa jo prezre. Spletna mobilna aplikacija je dostopna preko mobilnega brskalnika naprave. Uporabniki do posodobitev aplikacij pridejo brez posredovanja, saj so vsi na isti različici. Aplikacije je težje najti, saj ni drugih spletnih trgovin za prenos aplikacij, kot tista, ki je ponujena pri avtohtonih mobilnih aplikacijah.

	Avtohtona aplikacija	Spletna mobilna aplikacija
Jezik	Vsaka platforma svoj jezik	Spletne tehnologije
Dostop	Na mobilni napravi	Spletni brskalnik
Posodobitve	Ročno prenesti in namestiti Prezreti	Brez posredovanja Vsi z isto različico
Prednosti	Hitrejši Lažje dostopni in vidni drugim Zaslužek s plačljivimi vsebinami	Skupna koda Preurejanje s spletnim oblikovanjem Brez prenosa aplikacije Brez prenosa posodobitev Zaslužek z oglasi
Slabosti	Dražji razvoj Večji stroški vzdrževanja Prenosljivost na druge platforme Zahteve za spletno trgovino in omejitve	Omejen dostop do funkcij naprave Več vrst brskalnikov, večji stroški Težje najti aplikacijo Večje strokovno znanje razvijalca Omejenost pri dostopu do senzorjev

Tabela 2.1: Kratka primerjava avtohtonih in spletnih mobilnih aplikacij.

Razvoj spletne in avtohtone aplikacije je bistveno drugačen, kar je prikazano v tabeli 2.1. Razvidno je, da razvoj obeh ni ravno poceni. Ker govorimo o mobilni platformi in ker spletni razvoj ne dopušča uporabe nekaterih senzorjev (predvsem strojnih), je za to vrsto razvoja bolj primerna avtohtona aplikacija.

2.3 Razvoj platformno neodvisnih aplikacij

Ker učenje dodatnega programskega jezika vzame preveč časa, se v zadnjih nekaj letih razvija s pomočjo orodij, ki omogočajo uporabo iste kode na različnih mobilnih platformah. Napisano kodo se ustrezno prevede in namesti na več različnih operacijskih sistemov. Navedeno se imenuje orodje za razvoj platformno neodvisnih aplikacij, kjer je potrebno poznati le razvojni jezik, ki ga orodje zahteva. Pri nekaterih pa je to tudi orodje za razvoj. Na trgu je danes veliko ponudnikov orodij, ki se med seboj razlikujejo po razvojnem jeziku, podpori platforme, ceni itd.

Vedno večji razvoj mobilnih naprav teži k večjemu razvoju različnih operacijskih sistemov. Tako se v zadnjem času razvija vse več orodij, ki omogočajo razvoj aplikacij za več platform hkrati. Orodja se med seboj razlikujejo. V nadaljevanju si bomo ogledali tri orodja za platformno neodvisen razvoj, njihove glavne značilnosti in uporabo.

2.3.1 PhoneGap

PhoneGap podpira sedem mobilnih platform (iOS, Android, BlackBerry, Palm webOS, Windows Mobile, Bada in Symbian). Z njim izdelujemo hibridne mobilne aplikacije. Za razvoj aplikacij uporabljamo spletne tehnologije, skupek programskih jezikov za razvoj spletnih aplikacij (HTML, CSS in JavaScript) [6].

Aplikacije, razvite z orodjem PhoneGap, so podobne spletnih aplikacijam, saj tečejo v oknu brskalnika na mobilni napravi. Izbira brskalnika za mobilno platformo je različna. Razvoj aplikacije tako ne poteka tako hitro, saj je potrebno napisati različen CSS za vsak brskalnik. PhoneGap omogoča dostop do različnih senzorjev naprave (npr. kompas, senzor nagiba, lokacija) kot tudi do drugih funkcionalnosti telefona (npr. imenik, kamera, spomin, avdio in video vsebine). Zanimiva je tudi izdelava aplikacije za izbrano platformo, saj moramo upoštevati vsa pravila za izdelavo aplikacije na tej platformi in

izdelati osnovni začetni projekt. Sam PhoneGap ne nudi okolja za razvoj aplikacije. Za vsako platformo se uporablja ustrezno okolje. To pomeni, da se za Android uporablja Eclipse, za iOS Xcode, za Windows Phone pa Visual Studio. Komunikacija in nadzor nad napravo se izvajata v domorodnem jeziku, kot so npr. Java, Objektni C, C# ... Izdelavo mobilnih aplikacij olajša znanje spletnih tehnologij in uporaba tehnologij in knjižnic, kot so npr. PhoneGap API, UI Framework ... Za dostop do funkcionalnosti naprave se uporabljajo že napisane knjižnice. Obstajajo knjižnice za dostop do imenika, kamere, senzorja nagiba, do omogočanja aplikacije za snemanje in predvajanje zvočnih in video posnetkov, opozarjanje na nova obvestila (zvok, vibracija ...). Z letom 2012 je PhoneGap izdal storitev PhoneGap Build. Slednja izdelano aplikacijo prevede za določeno platformo. Deluje brez nameščenega SDK-ja. Ta storitev je plačljiva glede na število izdelanih privatnih aplikacij. Višina zneska je odvisna tudi od velikosti aplikacije. Slabost tega orodja je tudi, da sta pri prenosu aplikacije na drugo platformo potrebna optimizacija in testiranje.

Slika 2.4: Prikaz arhitekture orodja PhoneGap [7].

Kot je razvidno s slike 2.4., so razvijalcu na voljo tehnologije, ki jih lahko uporablja pri razvoju aplikacije (zgornji del slike). Povezavo med razvijalčevo kodo JavaScript in funkcionalnostmi naprave zagotavljajo vmesniki PhoneGap API (ang. *Application programming interface*) in ogrodje uporabniškega vmesnika, kot je na primer JQueryMobile. V spodnjem delu slike so prikazane nekatere funkcionalnosti naprave, do katerih ima ta dostop.

2.3.2 Titanium Appcelerator

Medplatformsko ogrodje Titanium se uporablja za razvoj mobilnih, tabličnih in namiznih aplikacij. Omogočen je razvoj za platforme iOS, Android in BlackBerry s pomočjo spletnih tehnologij [8]. Ogrodje obstaja v brezplačni in plačljivi različici. Pri uporabi brezplačne različice smo zelo omejeni. Izdelamo lahko le osnovno mobilno aplikacijo s pomočjo videoposnetkov in podpore skupnosti. Plačljiva različica pa omogoča naprednejši API ter pomoč razvijalcem, ki jo nudi strokovna ekipa.
Titanium deluje kot povezava med avtohtonim operacijskim sistemom in kodo razvijalca [8]. Ima svoje okolje za razvoj, ki se imenuje Titanium Studio, in je podobno okolju Eclipse. Je hitro in enostavno za namestitev. Omogoča upravljanje s projekti, zagon aplikacije na simulatorju ali napravi in pošiljanje aplikacije v App Store ali Android Market. S pomočjo SDK-ja prevede napisano kodo v domorodno kodo vsake platforme.

Kot je razvidno s slike 2.5, je na dnu operacijski sistem za odjemalce (Android, iOS ali mobilna spletna aplikacija). Na vrhu je aplikacija, razvita z JavaScript. Vmesni del pa predstavljata Titanium SDK in API. Titanium SDK sprejme zahteve, kot so npr. risanje gumbov, zagon fotoaparata in drugo. Vse to poda razvijalec v kodi.

Titanium ima zelo dobro podano dokumentacijo in forum. Z registracijo lahko vsak razvijalec sodeluje v forumu z vprašanji in odgovori. Tako si razvijalec hitreje poišče rešitev za težavo pri razvoju ali pa sam pomaga reševati probleme drugih. Slabša stran tega orodja pa je, da so nekateri moduli plačljivi (npr. delo z grafiko). Poleg tega je aplikacijo za iOS mogoče

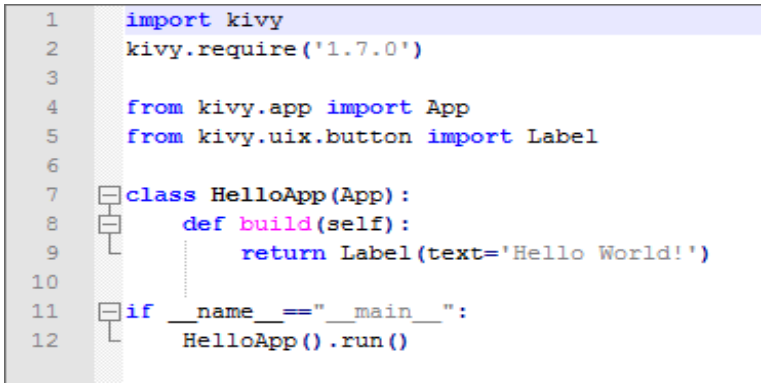
Slika 2.5: Arhitektura orodja Titanium [9].

razvijati le v Mac OS-u.

2.3.3 Kivy

Kivy je knjižnica za izdelavo platformno neodvisnih aplikacij. Za razvoj aplikacij uporablja jezik Python. Poleg razvoja mobilnih aplikacij ponuja tudi razvoj računalniških namiznih aplikacij. Kivy teče na platformah Linux, Windows, OSx, Android in iOS. Ista koda se lahko zažene na vseh podprtih platformah. Lahko se uporabi večina izvornih vhodov, protokolov in naprav, vključno z WM_Touch, WM_Pen, MAC OS X Trackpad, Magic Mause, Mtdev, Linux Kernell HID, TUIO in drugimi [10]. Podprta je tudi uporaba večdotične tehnologije.
Razvoj aplikacije lahko poteka v beležnici, ki nudi podporo programskim jezikom, ali pa se namesti orodje PyScripter oziroma drugo podobno orodje. Do vseh funkcionalnosti naprave aplikacija dostopa preko knjižnic. Pomembne knjižnice so PyGame, PIL, GStream, OpenGl in ostale. Sam začetek je lepo

opisan na spletni strani www.kivy.org, kjer se nahaja dokumentacija s primeri projektov in vaj. Navedena spletna stran nudi tudi forum s svežo in staro vsebino novic in člankov ter razne vodiče. Knjižnica Kivy je za uporabo brezplačna. Na sliki 2.6 je predstavljena koda Kivy v orodju PyScripter. Koda predstavlja program "Hello World".



```
1  import kivy
2  kivy.require('1.7.0')
3
4  from kivy.app import App
5  from kivy.uix.button import Label
6
7  class HelloApp(App):
8      def build(self):
9          return Label(text='Hello World!')
10
11 if __name__=="__main__":
12     HelloApp().run()
```

Slika 2.6: Koda za program "Hello World", napisana v jeziku Kivy.

2.3.4 Primerjava orodij

V tabeli 2.2 so predstavljena tri orodja in kriteriji, po katerih se ta razlikujejo med seboj. PhoneGap in Titanium (v primerjavi s Kivy) delujeta na osnovi spletnih tehnologij. Do senzorjev naprave dostopata preko vmesnika JavaScript API. Programski del oziroma logika teče znotraj domorodnega sistema. Kivy za svoj razvojni jezik uporablja Python, ki je preprost za uporabo. Za dostop do senzorjev pa uporablja knjižnice znotraj Pythona. PhoneGap podpira več mobilnih platform, Kivy pa poleg teh podpira tudi razvoj namiznih aplikacij. Izgled Titanium aplikacije je bolj domoroden kot izgled PhoneGap aplikacije. Appcelerator Titanium edini nudi svoje okolje za razvoj, PhoneGap aplikacijo pa razvijemo v okolju Eclipse IDE po istih postopkih kot izvorno aplikacijo.

	Kivy	PhoneGap	Appcelerator Titanium
Podprte platforme	Android, iOS, Windows Phone, Linux, Windows, OSx	Android, iOS, Windows Phone, BlackBerry, Symbian, Tizen, Bad	Android, iOS, Windows Phone, BlackBerry, Tizen
Razvojni jezik	Python	HTML, CSS, JavaScript	HTML5, CSS, JavaScript
Prednosti	Enostaven, podpora Python 3.0 Razvoj namiznih aplikacij Podpora za Raspberry Pi	Hiter in lahek dostop do kamere, kontaktov, GPS-a Spletne storitve	Svoje okolje za razvoj, lahka in hitra namestitev, uporaba vmesnikov Titanium, aplikacije so hitre in odzivne, dokumentacija in forum z vprašanji in odgovori
Slabosti	V določenih primerih je Python počasen, ni podpore za mikrofon, brez emulatorja	Za vsako platformo je potrebno uporabiti ustrezno okolje za razvoj, testiranje in optimiziranje za druge platforme	Aplikacijo za iOS je mogoče razvijati samo v Mac OS-u. Plačljivi moduli
Izdelava mobilnih aplikacij	Enostavno, z uporabo knjižnic	Znanje spletnih tehnologij, uporaba tehnologij	Uporaba modulov, podani primeri s ključnimi elementi

Tabela 2.2: Primerjava treh orodij za razvoj.

Poglavje 3

Kivy

Kivy je odprtokodna Python knjižnica za hiter razvoj aplikacij. Omogoča enostavno uporabo sodobnih uporabniških vmesnikov, za širok nabor naprav.
Za implementacijo kode ne potrebujemo posebnega razvojnega okolja, zadostuje že urejevalnik besedil. Podpira razvoj aplikacij za trenutno aktualne platforme – že omenjene Linux, Windows, OSx, Android in iOS. Aplikacije na platformah pa delujejo kot samostojne. V primerjavi z drugimi orodji za izdelavo platformno neodvisnih aplikacij, ki jih je danes zelo veliko, Kivy nudi dobro dokumentacijo s primeri. Prav tako nudi tudi forum s svežo in staro vsebino novic in člankov ter vodičev. V določenih primerih pa je tudi slab (npr. počasen Python). Če bi v razvoju Android aplikacije uporabili Javo in Android SDK, bi bila učinkovitost izvajanja aplikacije veliko boljša, vendar bi s tem izgubili možnost prenosljivosti na druge platforme.

Kivy je za uporabo brezplačen, in sicer na podlagi licence MIT (za 1.8 in 1.7.2)[1] in licence LGPL 3[2] za starejše različice.

[1]Licenca MIT je licenca za brezplačno programsko opremo, izdana na MIT (ang. *Massachusetts Institute of Technology*). Omogoča ponovno uporabo lastniške programske opreme, če vse kopije licenčne programske opreme vključujejo kopijo licenčnih pogojev MIT [11].

[2]LGPL (ang. *Lesser General Public License*) je licenca za brezplačno programsko

Kivy je naslednik PyMT[3]. Sledi preprostemu cilju: enaka koda za vsako platformo. Uveden je bil na konferenci EuroPython 2011 kot Python framework, oblikovan za ustvarjanje naravnih uporabniških vmesnikov [12].

Za sam razvoj uporablja Python, enostaven in zelo razširjen jezik. Ker mobilne naprave podpirajo in ponujajo veliko senzorjev, je uporaba teh s strani Kivy omogočena. Za dostop do teh funkcij se uporablja modul Plyer, ki poenostavi uporabo. Več o tem v nadaljevanju.

Kivy je namenjen tudi razvoju aplikacij za namizne računalnike. Za njihov razvoj uporablja knjižnice preko Pythona. Ta ima različne sistemske knjižnice, preko katerih se lahko dostopa do miške, ekranov na dotik, igralnih konzol ipd. Nekaj pomembnih knjižnic:

- PyPy: implementacija Pythona za pospešitev izvajanja programa. To doseže tako, da Python kodo prevede v strojno kodo računalnika.

- PyGame: set Python modulov za razvijanje iger. Vsebuje knjižnice za grafiko in zvok, ki jih lahko pokličemo iz kode v Pythonu. Lahko se uporablja na napravah Android, saj omogoča uporabo zvoka, vibracij, tipkovnice, kot tudi senzorjev nagiba. Žal se ga na sistemu iOS ne da uporabljati.

- PIL (ang. *Python Imaging Library*): Python knjižnica, ki omogoča obdelavo slik in kot taka podpira veliko formatov. Sem spadajo formati PPM, PNG, JPEG, GIF, TIFF, BMP in ostali. PIL ponuja več standardnih postopkov za obdelavo slik, kot so filtriranje slik, izboljšanje slik z ostrenjem, prilagajanje svetlosti, kontrasta in barv, dodajanje slik in še več [13].

opremo, ki jo je objavila FSF (ang. *Free Softwere Foundation*). Razvijalcem in podjetjem omogoča uporabo in integracijo programske opreme pod licenco LGPL v svojo (tudi lastniško) programsko opremo. Različica 3 je bila objavljena leta 2007, skupaj s seznamom dodatnih dovoljenj [14].

[3]PyMT je odprtokodna knjižnica za razvoj večdotičnih aplikacij. Napisan je v Pythonu in uporablja številne knjižnice, kot so PyOpengl, Pygame, PyGST ... Omogoča hiter in enostaven razvoj.

- GStream: knjižnica za predvajanje, snemanje, obdelavo in oddajanje zvoka ali videa. Namenjen je vsem operacijskim sistemom, ki so grajeni na jedru Linux. Poleg GStreama se lahko uporablja tudi podobno knjižnico FFmpeg.

- CAIRO: knjižnica, ki nudi vektorsko grafiko za razvoj aplikacij. Zasnovana je tako, da uporablja strojno pospeševanje, kadar je to mogoče.

- OpenGL: knjižnica za prikazovanje 2D in 3D grafike. Običajno direktno komunicira z grafično kartico, s čimer dosežemo strojno pospešeno prikazovanje grafike. Lahko se uporablja na vseh operacijskih sistemih.

Aplikacijo lahko razvijamo na kateremkoli sistemu. Potrebno je imeti naložene nastavitve, ki jih zahteva Kivy. Za zagon na drugih sistemih obstajajo postopki priprave aplikacije za zagon, t. i. paket. Postopki so podrobno razloženi v dokumentaciji. V sistemih Linux, Windows in xOS to ni potrebno.
Za uspešno razvijanje aplikacije Kivy je dobro vedeti, katere vmesnike API Kivy ne podpira in kako do njih dostopamo. Prav tako je zaželeno poznavanje možnosti, ki jih pri testiranju podpira simulator in ali je bolje testiranje opraviti na sami napravi.

Pri tem moramo poznati, kaj je skupno napravam, za katere razvijamo, in kako delujejo emulatorji. Dobro je vedeti tudi, kako v različnih primerih razvoja dostopati do API-jev.

Plyer

Plyer je implementiran predvsem za razvoj mobilnih aplikacij, kjer se uporablja za dostop do senzorjev naprave in drugih funkcionalnosti. Tabela 3.1 prikazuje strojno opremo naprave, ki jo lahko uporabimo pri razvoju aplikacije za določeno platformo s pomočjo Plyerja. Razvidno je, da v veliki večini podpira strojno opremo mobilnih platform, namizne le v manjšem obsegu. Plyer je neodvisen od platforme. V primeru Android uporablja Pyjnius. To je modul Python, ki omogoča, da dostopamo do Java razredov znotraj Pythona.

V tem primeru pomeni, da dostopamo do Android SDK-ja in NDK-ja. Za sistem iOS uporablja Pyobjus, ki je prav tako modul Python. Omogoča pa dostop do razredov Objective-C.

	Android	iOS	Windows	OS X	Linux
Merilnik pospeška	X	X		X	X
Kamera	X	X			
GPS	X	X			
Obvestila	X		X	X	X
E-pošta	X	X	X	X	X
Vibrator	X	X			
SMS (pošiljanje sporočil)	X				
Kompas	X	X			
Giroskop	X	X			
Baterija	X	X	X	X	X
Snemanje zvoka	X				
Bliskavica	X	X			

Tabela 3.1: Podpora API-jev na različnih platformah [15].

Android in simulator iOS

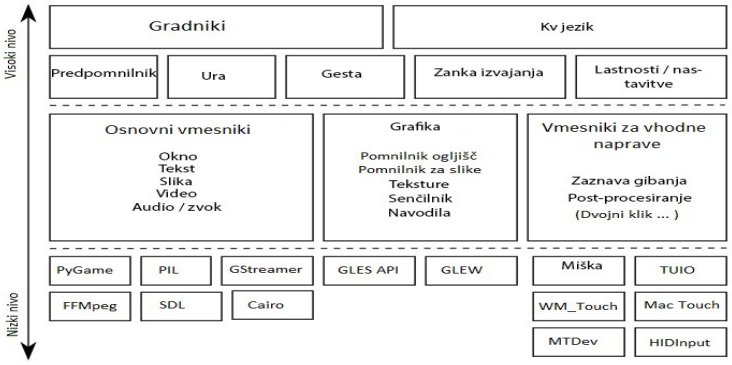
Uporaba simulatorja za testiranje aplikacije iOS se izkaže za neprimerno. Simulator, ki je del razvojnega okolja xCode, ne podpira celotne strojne opreme. Uporaba kamere, mikrofona, barometra, senzorja bližine, senzorja svetlobe in merilnika pospeška ni možna [16]. Prav tako simulator ne ponuja vseh funkcij, ki so dostopne na napravi, npr. prejemanje in pošiljanje obvestil podjetju Apple, opozorila o zasebnosti za dostop do fotografij, imenika, koledarja, dostop do zunanje dodatne opreme, do Media Playerja [16] ... V Android simulatorju (ang. *Android virtual device*) je testiranje aplikacije bolj primerno, saj podpira to strojno opremo.

3.1 Knjižnice

Kivy je odprtokodna knjižnica za razvoj aplikacij. Zasnovana je tako, da omogoča hiter in enostaven zapis interaktivne aplikacije. Obstajajo visokokakovostne knjižnice, ki jih lahko vključimo v aplikacijo. Hkrati pa zmogljive kritične dele izvaja v jeziku C. Nekaj pomembnih knjižnic:

- kivy.graphic: ponuja, kar potrebujemo za risanje. Osnovni razred je Canvas, ki predstavlja platno za risanje. Celoten grafični paket podpira OpenGL.
- kivy.modules: so razredi, ki se lahko naložijo ob zagonu aplikacije Kivy. Aktiviramo jih lahko na tri načine: v datoteki config.ini, v samem Python programu (primer: from kivy.config import Config ...), ali pri zagonu ukaza Python. Lahko ustvarimo tudi svoje module.
- kivy.UIX: vsebuje gradnike za grafično okolje (tj. gumb, kamera, postavitev, izbor barve, izbira polja).
- kivy.language: definira jezik, ki opisuje uporabniški vmesnik in interakcijo. Ima pravila, ki so podobna CSS (prekrivni slogi).
- kivy.event: vsebuje vse za upravljanje z dogodki. Dogodek se lahko sproži ali posluša.
- kivy.input: ponuja vse za sprejemanje dogodkov iz različnih vhodov.

3.2 Arhitektura Kivy



Slika 3.1: Arhitektura orodja Kivy [17].

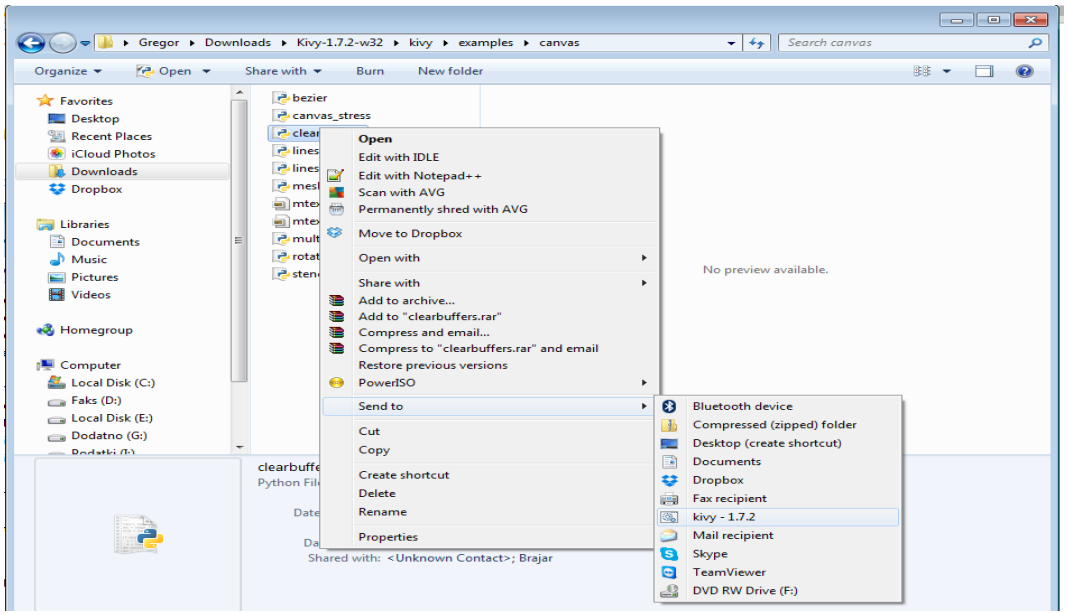
Slika 3.1 prikazuje sloje arhitekture Kivy. Delijo se na višji in nižji nivo in so sestavljeni iz več blokov. V višji nivo spada jezik KV ter gradniki uporabniškega vmesnika in postavitve, v nižji nivo pa specifični avtohtoni API-ji operacijskega sistema. Vmesni del, med nižjim in višji nivojem, pa sestavljajo trije vmesniki, ki skrbijo za povezavo med njima. Za povezavo med specifičnimi avtohtonimi API-ji operacijskega sistema in arhitekturo Kivy skrbijo osnovni vmesniki (ang. *Core providers*). Ti vmesniki opravlja osnovne naloge (ang. *Core task*). Cilj arhitekture Kivy je, da so te naloge abstraktne in modularne, kar pomeni, da so enostavne za uporabo in razširljive. Isti princip uporablja vmesnik za vhodne naprave (ang. *Input provider*). Ta vmesnik je del kode, ki skrbi za podporo vhodnih naprav, kot so simulator miške, ekran in podobno. Za uporabo nove vhodne naprave napišemo razred, ki bere vhodne podatke naprave in jih posreduje arhitekturi Kivy. V vmesnem delu se nahaja tudi grafika, ki je pravzaprav abstrakcija OpenGL, saj preko nje Kivy uporablja strojno podprte ukaze z uporabo OpenGl-a. Dostopa do knjižnic na nižjem nivoju. Omogoča uporabo zahtevnih in matematično kompleksnih funkcij OpenGl na enostaven način. Tako razvijalcu ni potrebno podrobno poznati OpenGl-a. Vmesnik je napisan v jeziku C, kar zagotavlja hitro delovanje. Skrbi za avtomatsko optimizacijo funkcij za

risanje. Koda je zato bolj učinkovita.

3.3 Razvoj s Kivy

3.3.1 Namestitev

Kivy je knjižnica, ki za ustvarjanje aplikacij za mobilne naprave in namizne računalnike omogoča uporabo jezika Python. Namestitev poteka za vsak operacijski sistem različno. Vsa potrebna navodila so po korakih zapisana v dokumentaciji. Pri Windows in MAC OS je namestitev podobna in nekomplicirana.



Slika 3.2: Način zagona napisane kode v Pythonu.

Slika 3.2 prikazuje, kako zagnati že napisano datoteko Kivy v okolju Windows. Za zagon je pripravljena skripta, preko katere se program izvede.

Poleg začetne namestitve obstajajo tudi neobvezni programski paketi:

- OpenCu 2.0 za kamero

- PIL in PyCairo za prikaz slik in teksta na zaslonu
- PyGST za avdio in video predvajanje ter za uporabo kamere

3.3.2 Postavitev v Kivyju

Razvoj v orodju Kivy je razdeljen na dva dela in ločuje predstavitev od logike. V prvem delu z jezikom Kivy opišemo strukturo uporabniškega vmesnika. Lahko je podan v posebni datoteki z istim imenom kot ime razreda (brez besede App) in s končnico .kv ali pa v isti datoteki pred razredom. Napisano je v jeziku Kivy in se zažene samodejno. Drugi del pa sestavlja koda aplikacije, zapisana v jeziku Python.

Za razvoj aplikacije ni potrebno nameščanje orodja, dovolj je že urejevalnik. Lahko pa se seveda uporabi orodje za programiranje v jeziku Python, npr. PyScripter. Na začetku je treba definirati različico Kivyja, s katerim bomo delali.

```
BoxLayout:
    FloatLayout:
        id: fl
    Button:
        text: 'Izberi sliko'
        on_press: app.show_load()
        size_hint_y: None
        height: '48dp'
    Button:
        text: 'Kamera'
        on_press: app.prikaziKamero()
        size_hint_y: None
        height: '48dp'
<LoadDialog>:
    BoxLayout:
        size: root.size
        pos: root.pos
        orientation: "vertical"
        FileChooserListView:
            id: filechooser

        BoxLayout:
            size_hint_y: None
            height: 30
            Button:
                text: "Cancel"
                on_release: root.cancel()

            Button:
                text: "Load"
                on_release: root.load(filechooser.path, filechooser.selection)
```

Slika 3.3: Zapisana struktura uporabniškega vmesnika v posebni obliki Kivy.

Slika 3.3 prikazuje strukturo uporabniškega vmesnika. Posebna koda, ki je na njej, se imenuje Kivy. Jezik Kivy je preprost in namenjen opisovanju uporabniškega vmesnika. Obstajajo številne postavitve in pripomočki, ki

bodo opisani v nadaljevanju. Sama struktura spominja na jezik CSS in na slog za HTML.

Z uporabo gradnikov grafično predstavimo uporabniški vmesnik (UV). Na voljo so osnovni in kompleksni gradniki. Ti gradniki so: oznaka (ang. *Label*), gumbi (ang. *Button*), polja za vnos (ang. *Text Field*), stikala (ang. *Switch*), slika (ang. *Picture*), video (ang. *Video*), drsnik (ang. *Slider*), polja za izbiro (ang. *ListView*), seznami (ang. *List*), video predvajalnik (ang. *VideoPlayer*), tipkovnica (ang. *Keyboard*), zavihki (ang. *Tab*), krožno polje (ang. *Spinner*) itd. Za razporeditev gradnikov po zaslonu uporabimo posebno vrsto gradnikov, imenovanih postavitve [18]. Sem spadajo tako imenovane:

- Postavitev v škatle (ang. *BoxLayout*)
- Mrežna postavitev (ang. *GridLayout*)
- Postavitev v sklad (ang. *StackLayout*)
- Postavitev s sidrom (ang. *AnchorLayout*)
- Plavajoča postavitev (ang. *FloatLayout*)
- Relativna postavitev (ang. *RelativeLayout*)

Prav tako vsebuje tudi grafični gradnik platno. Ta nam omogoča uporabo objektov, kot so pravokotnik, krog, elipsa in ostalo [19]. Dostopni so preko knjižnic. Gradnikom lahko določimo tudi atribute (tekst, velikost teksta, pozicija, orientacija, poravnava, akcija, barva, id ...) [20]. Vse te atribute se dodaja s pomočjo knjižnice kivy.graphic [21]. V primeru, da imamo več gradnikov z istimi lastnostmi, lahko gradnik določimo kot globalno spremenljivko. Ostali gradniki pa podedujejo te lastnosti. Za oblikovanje uporabniških vmesnikov je na voljo tudi orodje Kivy Designer [22, 23]. Z njim lahko gradnike ustvarimo, prilagodimo in preizkusimo. Zagotavlja ustvarjanje vmesnika za namizno aplikacijo, aplikacijo Android in aplikacijo iOS.

Poglavje 4

Testna aplikacija

Danes je kamera nepogrešljiv del mobilne naprave. Veliko aplikacij zajema sliko s kamero. Sliko se potem poljubno ureja ali pošlje drugemu uporabniku. Iste aplikacije najdemo na mobilnih sistemih Android, iOS in Windows. Za svojo diplomsko nalogo sem ustvaril aplikacijo, ki zajame sliko s kamere oziroma jo naloži preko galerije. Slika se prikaže na zaslonu in jo lahko poljubno obračamo, povečujemo, zmanjšujemo in premikamo.

4.1 Opis funkcionalnosti mobilne aplikacije

Začetni grafični prikaz vsebuje dva gumba v postavitvi v škatle. Dodana sta na dnu zaslona. Funkciji, ki ju opravljata, sta izbira slike iz galerije in zagon kamere. Za vsako funkcijo smo potrebovali svoj grafični prikaz.

4.1.1 Dodajanje slike na zaslon

Sliko lahko na zaslonu prikažemo na dva načina. S prvim načinom izberemo že shranjeno sliko, ki jo najdemo v pomnilniku naprave. Drugi način pa omogoča, da sliko zajamemo s kamero in jo prikažemo na zaslonu.

4.1.2 Interakcije nad slikami

Da je vse skupaj bolj zanimivo, dodani sliki omogočimo, da jo lahko poljubno obračamo, povečujemo in premikamo.

4.2 Prenos aplikacije na mobilno napravo

Android

Aplikacijo, ki je razvita v okolju Kivy, se po posebnih postopkih naloži na mobilno napravo. Za uvoz na napravo Android obstajata dve opciji. Če ne razvijamo na sistemu Linux, je priporočljiva uporaba že pripravljenega virtualnega orodja, kot je na primer Buildozer.

Buildozer

Buildozer je orodje, ki iz pripravljene kode ustvari paket. Ta avtomatizira celoten proces ustvarjanja paketa. Prvo ustvarjanje je dolgo, saj je potrebno prenesti vse predpogoje, kot so Python-for-Android, Android SDK, NDK in druga orodja, ki jih potrebuje za sestavo paketa. Nato ustvari datoteko buildozer.spec, v kateri opiše vse zahteve aplikacije, nastavitve, naslov, ikone, vključene module itd [24]. V datoteki se definira zahteve mobilnega operacijskega sistema, od SDK-ja, NDK-ja, API-jev, do verzije in ostalih dovoljenj. Trenutno nudi podporo za ustvarjanje paketa za Android ter za sistem iOS. Za slednjega je potrebno imeti računalnik s sistemom OSx. V prihodnje orodje Buildozer načrtuje podporo tudi drugim platformam (.exe za Windows, .dmg za OSX) [24].

Drugi, nekoliko daljši način pa zahteva, da se celotno ustvarjanje paketa ustvari postopoma, za kar je potreben paket Python-for-Android. Če je mobilna naprava povezana z računalnikom, se po končanem postopku aplikacija avtomatično zažene na telefonu. V nasprotnem primeru se ustvari ustrezen paket, npr. za sistem Android paket .apk.

IOS

Za pripravo paketa na operacijskem sistemu iOS je potreben računalnik s sistemom OSx in orodjem XCode. Pomembna je tudi verzija XCoda. Verzija 7 omogoča brezplačno testiranje aplikacije na sistemu iOS. Do te verzije je potrebna registracija Apple ID-ja na Apple Developer in plačilo letne članarine.

4.3 Razvoj aplikacije

4.3.1 Grafični prikaz uporabniškega vmesnika

V prvi fazi razvoja smo izdelali uporabniški vmesnik s posebnim jezikom, ki se imenuje Kivy. V datoteki .kv se na začetku definira različico Kivyja. V nadaljevanju se za vsak primerek razreda izdela blok, ki je znotraj <> predstavljen z imenom razreda. Posameznemu gradniku lahko dodamo tudi odzivno funkcijo, ki se izvede ob določenem dogodku.

```
kv = '''
<Kamera>:
    id: cam
    orientation: 'vertical'

    Button:
        text: 'Slikaj'
        on_press: cam.slikaj()
        size_hint_y: None
        height: '30dp'

BoxLayout:
    FloatLayout:
        id: fl
    Button:
        text: 'Izberi sliko'
        on_press: app.show_load()
        size_hint_y: None
        height: '48dp'
    Button:
        text: 'Kamera'
        on_press: app.prikaziKamero()
        size_hint_y: None
        height: '48dp'

<LoadDialog>:
    BoxLayout:
        size: root.size
        pos: root.pos
        orientation: "vertical"
        FileChooserListView:
            id: filechooser
```

Slika 4.1: Del implementacije postavitve UV.

Slika 4.1 prikazuje del kode .kv, ki prikaže začetno postavitev uporabniškega

vmesnika. S slike je razvidno, da je uporabljena tako imenovana postavitev v škatle ali BoxLayout. Gumbom je dodana akcija, ki pove, katera metoda se ob pritisku na gumb izvede. Ker je v kodi Python definiran razred "LoadDialog", se v datoteki .kv definira nov blok z istim imenom. Nov razred je zato, ker se s prikazom dialoga za izbiro datoteke prikaže nov zaslon.

4.3.2 Povezava logike in postavitve skupaj

Slika 4.2 prikazuje implementacijo glavnega razreda. S klicem metode Builder.load_string(kv) preberemo strukturo, ki smo jo zgradili z jezikom KV, in sestavimo osnovni uporabniški vmesnik (UV). Prav tako je potrebno implementirati metode, ki so potrebne za delovanje osnovnega uporabniškega vmesnika. V tem primeru sta na razpolago gumba "Kamera"in "Izberi sliko"(glej sliko 4.1), ki ob aktivaciji pokličeta metodi prikaziKamero() in show_load().

```
# Glavni razred
class TestCamera(App):

    # Metoda ki zgradi aplikacijo in inicializira razlicne elemente uporabniskega vmesnika
    def build(self):
        Logger.info('Test Camera Init')
        # Prebere strukturo definirano zgoraj in iz nje sestavi osnoven uporabniski vmesnik
        self.root = Builder.load_string(kv)


        # Inicializacija prikaznega okna za izbiro datoteke pri izbiri slike
        content = LoadDialog(load=self.load, cancel=self.dismiss_popup)
        self._popup = Popup(title="Load file", content=content,
                            size_hint=(None, None),size=(400,400))

        return self.root

    # Metoda za prikaz kamere
    def prikaziKamero(self):
        camera.take_picture("/storage/sdcard0/DCIM/slika.jpg",self.done)

    def done(self, e):
        Clock.schedule_once(partial(self.load,'/storage/sdcard0/DCIM',['slika.jpg']), 0)

    # Metoda za prikaz okna, za izbiro datotek
    def show_load(self):
        self._popup.open()

    # Metoda za odpiranje izbrane datoteke (klicana iz prikaznega okna kamere)
    def load(self, path, filename, *args):

        Cache.remove('kv.texture')
        Cache.remove('kv.image')

    pic=Picture(source = os.path.join(path, filename[0]),rotation=randint(-25,25),nocache=True)
```

Slika 4.2: Realizacija prikaza kamere in shranjevanje slik na napravo.

V primeru, ki ga prikazuje slika 4.1, se gradnik gumb (Button) odzove na akcijo pritisni (on_press), s klicem funkcije cam.slikaj(). Lastnost nam pove,

katera metoda se sproži ob pritisku na gumb. S tem povežemo predstavitev in logiko.

4.3.3 Primer uporabe aplikacije

Slika 4.3: Primer prikaza aplikacije na tabličnem in mobilnem sistemu Android.

Slika 4.3 prikazuje delovanje aplikacije na dveh različnih napravah Android. Ista koda pokaže isti rezultat na mobilni napravi in na tabličnem računalniku. Potek namestitve je pri obeh napravah enak, prav tako tudi sama uporaba. Isto aplikacijo smo zagnali na sistemu iOS in ni delala večjih težav. Optimizacija ni bila potrebna, temveč smo le ustrezno prilagodili kodo.

4.3.4 Testiranje

Namestitev aplikacije na obe mobilni platformi je potekala s pomočjo priložene dokumentacije [25, 26]. Namestitev za mobilno platformo iOS je potekala na namiznem računalniku s sistemom OSx in programom xCode. Z omenjenim programom smo namestili aplikacijo na mobilno napravo. Aplikacija je na obeh sistemih delovala enako, tako da optimizacija ni bila potrebna. V sami razvojni kodi pa je bilo potrebno opraviti manjše prilagoditve. Prav tako smo testirali aplikacijo na tabličnem računalniku s sistemom Android. Tudi tu ni bilo potrebe po optimizaciji.

Poglavje 5

Sklepne ugotovitve

Nit diplomske naloge je tekla o mobilnem medplatformskem razvoju. Ta je danes ključnega pomena, saj mobilne aplikacije v zadnjih letih prevzemajo vse večji delež uporabnosti. V nalogi so predstavljene razvojne metode, njihove prednosti in slabosti. Do danes je razvitih kar nekaj orodij za razvoj platformno neodvisnih aplikacij. Predstavil sem dve bolj poznani orodji in eno manj, na katerem je temeljil drugi del diplomske naloge. Danes mobilne naprave ponujajo vrsto podobnih funkcionalnosti, zaradi katerih je tudi medplatformski razvoj smiseln. Namen te naloge je bil spoznavanje orodja s primeri in opisi razvoja v njem.

Pri razvoju primera je bilo ugotovljenih kar nekaj pomanjkljivosti. Občutek sem imel, da razvijalci Kivyja niso do potankosti razvili določenih stvari. Pri preprostih primerih sem naletel na kar nekaj hroščev (ang. Bug), za katere sem porabil kar nekaj časa, da sem poiskal rešitev. Če bi nam bila pomembna učinkovitost izvajanja aplikacije, bi bile avtohtone aplikacije boljša izbira. Vendar potem izgubimo možnost prenosljivosti na druge platforme. Velika prednost razvoja s knjižnico Kivy je ta, da razvoj poteka v jeziku Python, ki omogoča dostop do javanskih razredov (v primeru aplikacij za Android). Tu je namreč enostavno ločiti postavitev uporabniškega vmesnika od kode.

Pri razvoju aplikacije je bilo ugotovljeno, da je Kivy brez dodatnih modulov neuporaben. Del diplomske naloge je bil tudi razvoj operacij nad objektom (v našem primeru nad slikami). Izbira modula kivy.uix.scatter je bila prava izbira. Velika prednost razvoja z orodjem za medplatformski razvoj je čas razvoja, ki je mnogo krajši, ter cenejše in lažje vzdrževanje.

Literatura

[1] Native mobile App [Online]. Dosegljivo: http://www.techopedia.com/definition/27568/native-mobile-app [Dostopano: 23.10.2015].

[2] Priya Viswanathan (2015) The Pros and Cons of Native Apps and Mobile Web [Online]. Dosegljivo: http://mobiledevices.about.com/od/additionalresources/qt/The-Pros-And-Cons-Of-Native-Apps-And-Mobile-Web-Apps.htm [Dostopano: 23.10.2015].

[3] Web sensor API: raw and uncut [Online]. Dosegljivo: http://smus.com/web-sensor-api/ [Dostopano: 28.10.2015].

[4] HTML5 compatibility on mobile and tablet browsers with testing on real devices [Online]. Dosegljivo: http://mobilehtml5.org/ [Dostopano: 28.10.2015].

[5] Sachin Datta (2015) Should you go native (or web, or hybrid)? [Online]. Dosegljivo: http://thinking.edynamic.net/should-you-go-native-or-web-or-hybrid-how-to-choose-the-right-mobile-app [Dostopano: 23.10.2015].

[6] M. Zimic. Medplatformski razvoj mobilnih aplikacij. Diplomsko delo, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2012 [Online]. Dosegljivo:

http://eprints.fri.uni-lj.si/1726/1/Zimic-1.pdf/ [Dostopano: 25.10.2015].

[7] Multi-Platform Mobile Apps with PhoneGap [Online]. Dosegljivo: http://themoderndeveloper.com/the-modern-developer/the-ux-client-series-phonegap-mobile-apps [Dostopano 29.10.2015].

[8] U. Jenko. Medplatformski razvoj mobilne aplikacije s podporo v oblaku. Diplomsko delo, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2013 [Online]. Dosegljivo: http://eprints.fri.uni-lj.si/2213/1/Jenko_U-1.pdf [Dostopano 30.10.2015].

[9] Cross platform native development with appcelerator titanium (2014 devnexus). [Online]. Dosegljivo: http://www.slideshare.net/stephenfeather/cross-platform-native-development-with-appcelerator-titanium-2014-devnexus [Dostopano 30.10.2015].

[10] Kivy Providers [Online]. Dostopano: https://kivy.org/docs/api-kivy.input.providers.html [Dostopano: 2.11.2015].

[11] MIT license [Online]. Dosegljivo: https://en.wikipedia.org/wiki/MIT_License [Dostopano: 15.1.2016].

[12] Kivy (Next PyMT) [Online]. Dosegljivo: https://kivy.org/planet/2011/01/kivy-next-pymt-on-android-step-1-done/ [Dostopano: 29.10.2015].

[13] Python Imaging Library [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Python_Imaging_Library [Dostopano: 29.10.2015].

[14] GNU Lesser General Public License [Online]. Dosegljivo: https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License [Dostopano: 15.1.2016].

[15] Plyer [Online]. Dosegljivo: https://github.com/kivy/plyer/ [Dostopano: 27.1.2016].

[16] Apple emulator [Online]. Dosegljivo: https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/Introduction/Introduction.html [Dostopano: 26.1.2016].

[17] Kivy Arhitecture [Online]. Dosegljivo: http://kivy.org/docs/guide/architecture.html [Dostopno: 31.10.2015].

[18] Widget – Organize with Layouts [Online]. Dosegljivo: https://kivy.org/docs/guide/widgets.html?highlight=background [Dostopano: 15.1.2016].

[19] Kivy Canvas [Online]. Dosegljivo: https://kivy.org/docs/api-kivy.graphics.instructions.html [Dostopano: 15.1.2016].

[20] Kivy Widgets [Online]. Dosegljivo: https://kivy.org/docs/api-kivy.uix.html [Dostopano: 15.1.2016].

[21] Graphics [Online]. https://kivy.org/docs/api-kivy.graphics.html [Dostopano: 15.1.2016].

[22] Kivy Designer [Online]. Dosegljivo: https://github.com/kivy/kivy-designer [Dostopano: 15.1.2016].

[23] Kivy Designer documentation [Online]. Dosegljivo: https://kivy-designer.readthedocs.org/en/latest/ [Dostopano: 15.1.2016].

[24] Buildozer [Online]. Dosegljivo: http://buildozer.readthedocs.org/en/latest/ [Dostopano 2.11.2015].

[25] Create package for Android [Online]. Dosegljivo: https://kivy.org/docs/guide/packaging-android.html [Dostopano: 15.1.2016].

[26] Create package for iOS [Online]. Dosegljivo: https://kivy.org/docs/guide/packaging-ios.html [Dostopano: 15.1.2016].